МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Искусственные нейронные сети»

Тема: «Генерация текста на основе “Алисы в стране чудес”»

Студент гр. 7381 _____ Лукашев Р.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Рекуррентные нейронные сети также могут быть использованы в качестве генеративных моделей.

Это означает, что в дополнение к тому, что они используются для прогнозных моделей (создания прогнозов), они могут изучать последовательности проблемы, а затем генерировать совершенно новые вероятные последовательности для проблемной области.

Подобные генеративные модели полезны не только для изучения того, насколько хорошо модель выявила проблему, но и для того, чтобы узнать больше о самой проблемной области.

Задачи.

- Ознакомиться с генерацией текста;
- Ознакомиться с системой Callback в Keras.

Требования.

1. Реализовать модель ИНС, которая будет генерировать текст;

2. Написать собственный CallBack, который будет показывать то как генерируется текст во время обучения (то есть раз в какое-то количество эпох генерировать и выводить текст у необученной модели);

3. Отследить процесс обучения при помощи TensorFlowCallBack, в отчете привести результаты и их анализ.

Ход работы.

Для исследования была выбрана предложенная в методических указаниях архитектура модели ИНС (см. рис. 1)

```
model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Рисунок 1 – выбранная архитектура модели.

Для отслеживания результатов в процессе обучения была написана функция, генерирующая текст, и Callback класс, использующий эту функцию в конце заданных пользователем эпох на текущей модели (см. рис. 2 и 3).

```
class MonitorCallback(Callback):
    def __init__(self, epochs):
        super().__init__()
        self.epochs = epochs

    def on_epoch_end(self, epoch, logs=None):
        if epoch in self.epochs:
            generate_text(epoch, model)
```

Рисунок 2 – собственный TensorflowCallback класс.

```
def generate_text(epoch, model_):
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
    sequence = []
    sys.stdout.write("Generated text - ")
    for i in range(1000):
        x = numpy.reshape(pattern, (1, len(pattern), 1))
        x = x / float(n_vocab)
        prediction = numpy.array(model_.predict(x, verbose=0))
        index = numpy.argmax(prediction)
        result = int_to_char[index]
        sequence.append(result)
        sys.stdout.write(result)
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    f = open("generated_text.txt", "a")
    f.write('On epoch ' + str(epoch) + ':\n' + ''.join(sequence) + '\n')
    f.close()
```

Рисунок 3 – функция генерации текста.

Сеть тренировалась на 20 эпохах, для проверочной генерации текста были выбраны эпохи 1, 2, 3, 5, 7, 10, 14, 15, 16, 17, 18, 19, 20. Далее посмотрим на самые интересные моменты. В местах, где встречается троеточие, оно означает, что последняя повторяющаяся фраза повторяется до конца сгенерированного текста.

On epoch 0:

tee the the the the the the the the the the the the the the the the the the …

За первую эпоху сеть выучила самое часто встречающееся слово, и сгенерировала текст, в котором оно много раз повторяется.

On epoch 1:

the and the and the and the and the and the and the and the …

Далее сеть выучила второе самое часто встречающееся слово, и научилась соединять слова. Слово and скорее всего было выучено не спроста, и, как мы сможем убедиться далее, уже сейчас сеть начинает понимать соединительное назначения этого слова.

On epoch 2:

' said the cateh to the catee aadin aadin 'nh the tone to the to the to the to the to the to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore to the tore …

Сеть потихоньку начинает учиться более разнообразно продолжать данный ей на вход текст, начинают появляться новые несуществующие слова. Появляется намек на выучивание сетью диалоговых конструкций.

On epoch 3:

the

Тут сеть сгенерировала одно слово и кучу пробелов, что показывает, что в начале обучения сеть очень чувствительна ко входным данным, и на редко

встречаемых необычных кусках текста начинает теряться, и выбирает самый не затратный на ресурсы путь генерации.

On epoch 5:

veo to tee to tee to tee to tee to tee to tee toe tas and the cad to the cad to the tas oo the cad to the tas oo the cad to the tas oo the cad to the tas oo the cad …

Сеть продолжает учиться составлять более длинные фразы.

On epoch 7:

o the toeee th the care and the was ho a lort oo the tar oo the tabte to the toree th the care and the was ho a lore to the tar oo the tabte to the toree th the care and the was ho a lort oo the tar oo the tabte to the toree th the care and the was ho a lore to the tar oo the tabte to the toree th the care and the was ho a lort oo the tar oo …

Сеть научилась комбинировать новые слова в длинные фразы за счет использования and. Слова становятся все длиннее и длиннее. Как можно заметить, слова состоят преимущественно из одних и тех-же букв. Также можно заметить, что эти буквы являются наиболее часто встречаемыми в языке.

On epoch 10:

a lotg ti tee the sooe '

'io tou te tee toe tait to bet, said the manter. 'i venl to the seet to bete the sooeo '

io sas toe tas a ait oo the coore

Сеть впервые поставила знак абзаца. В конце сеть сгенерировала кучу пробелов. Это произошло потому, что на вход попал кусок текста со стихотворением, и сеть быстро поняла, что знаки абзаца и подряд идущие пробелы есть хорошо. Также, как можем заметить здесь и далее, знаки абзаца стимулируют разнообразие генерируемых слов.

On epoch 14:

a little soreose toiee and the weit on whrh the cousd and the sooed io the ciul that it was a little coalesse that had soeee the dareen she was aolng the dia no the could the had nottle the sas a lintle garlen that she was aolng th the cous of the cousd and the was aolng the tai io the ciue thet sas an ince an the could and the sooed an ilre the rame then she was aolng th the cous of the cousd …

К этой эпохе сеть уже начала генерировать грамматически правильные фразы, хотя до сих пор не научилась пунктуации.

On epoch 15:

inre th the taid thing toene in the rooesss '

'io soet ho ' said the katter an an toee an the cad no the sabdi of the garden, and the white rabbit was toenk it was the white rabbit, and the white rabbit was toenk it was the white rabbit, and the white rabbit was toenk …

По версии данной ИНС, это был белый кролик, и белый кролик был тоенк. Сеть заучила пару новых слов и новую фразу, начала появляться пунктуация.

On epoch 16:

ve toen so the teat ' said the drypouse, "i d nott thin the wey of toine in the cirttrs!'

'i d nottle to tee that ' said the morke, "i d netely tooeo oo then '

'io s a liie tirt ho the cirtens, lhie the marthr war a lott oa toiee

"io toit hi the mant wire the wiite woul fh

* *

* *

* *

* *

* *

* *

Звездочки обозначают конец главы, начало новой главы, а также встречаются в некоторых местах текста. Когда они есть, их в тексте встречается много и подряд. На вход генератору попал кусок, содержащий звездочку, поэтому сеть решила практически за бесплатно сгенерировать кучу звездочек и пробелов. Также сеть к данной эпохе успела выучить и успешно применяет конструкцию “фраза” said the кто-то. Также не может не радовать наличие четкой пунктуации.

On epoch 17:

'i dan tou d aitt ane toese ' said the mock turtle in a sole of great ourelf,tone, "that sore to toue ' said the goyphon, and the teite rabbit was soe was oo tie toen saa io whs lad to tee the was of the was oo the tiete rabbit, and the wert on al incet oo the whs, whet hl was toe bila tone a little sotee the sabdit wote,
'i vhsu to the seat to tee that i mene'the gorsess iere ' said the mock turtle in a sore of great hurr,an the cage,
' *
* *
* *
* *
* *
*

Практически та-же ситуация, что и на предыдущей эпохе, но к “фраза” said the … добавляется описательный элемент in a каком-то чем-то (great hurr например). Т.е. сеть научилась описывать то, в какой манере разговаривают персонажи.

On epoch 18:
le taid to the career and then she had never bere the harter and then she was aoling

an in soee an fer oo the caree oi the tas of the gorseon, and then she had never bele that it was a lirtle oort or the sas of the gorseon, and then she was aelind the foore the had never dane and then she had heve the sai oo her aeai in the court, and the was aoling the fid to see thet sas oo the toodd the ramd tiie the had heve then she had heve the sai oo her aeai in the court, and the was aoling the fid to see thet sas oo the toodd the ramd tiie the had heve then she had heve the sai oo her aeai in the court, and the was aoling the fid to see thet sas oo the toodd the ramd tiie the had heve then she had heve the sai oo her aeai in the court, and the was aoling the fid to see thet sas oo the toodd the ramd tiie the had heve then she had heve the sai oo her aeai in the court, and the was aoling the fid to see thet sas oo the toodd the ramd tiie the had heve then she had heve the sai oo her aeai in t

Сеть уверенно использует выдуманные ею слова. Появилась новая конструкция and then …, а также просто эти два слова по отдельности уже точно обрели логическую значимость, что позволило сети генерировать длинные грамматически грамотные цепочки слов. Однако из-за этого почти полностью пропала пунктуация.

On epoch 19:

al * * * * * * * * * * * *

* *

* *

* *

* *

* *

* *

* *

* *

На последней эпохе при проверке мы настигли конец главы. Как мы выяснили раньше, и как видно сейчас, это довольно плохо, поскольку сеть выучила, что конструкцию “* ” можно повторять много раз подряд, и,

поскольку в тексте есть несколько длинных кусков, в которых эта конструкция встречается много раз подряд, степень похожести на текст остается большой несмотря на отсутствие разнообразия, присущего остальному тексту.

Выводы.

В ходе выполнения данной лабораторной работы была разработана и обучена модель, которая генерирует текст на основе обучающего материала, в качестве которого выступило произведение Льюиса Кэрролла «Приключения Алисы в стране чудес». В ходе эксперимента выяснилось, что по ходу обучения, генерируемый текст приобретает все больший и больший вокабулярий, хотя большинство слов являются вариациями реально используемых. Сеть способна запоминать и генерировать логические конструкции, и чем дольше сеть обучается, тем более грамматически правильные конструкции она использует.

Среди минусов этой сети можно выделить слишком большое время обучения (хотя размер тренировочных данных довольно большой, поэтому эта проблема присуща большому числу сетей), а также чрезмерное заучивание и стремление к повторению, из-за чего проявляется большая чувствительность к стартовым данным.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import sys
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint, Callback


filename = "wonderland.txt"
raw_text = open(filename).read()
raw_text = raw_text.lower()

chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))
int_to_char = dict((i, c) for i, c in enumerate(chars))

n_chars = len(raw_text)
n_vocab = len(chars)
print("Total Characters: ", n_chars)
print("Total Vocab: ", n_vocab)

seq_length = 100
dataX = []
dataY = []
for i in range(0, n_chars - seq_length, 1):
    seq_in = raw_text[i:i + seq_length]
    seq_out = raw_text[i + seq_length]
    dataX.append([char_to_int[char] for char in seq_in])
    dataY.append(char_to_int[seq_out])
n_patterns = len(dataX)
print("Total Patterns: ", n_patterns)

X = numpy.reshape(dataX, (n_patterns, seq_length, 1))
# normalize
X = X / float(n_vocab)
# one hot encode the output variable
y = np_utils.to_categorical(dataY)

model = Sequential()
model.add(LSTM(256, input_shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(y.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')


def generate_text(epoch, model_):
    start = numpy.random.randint(0, len(dataX) - 1)
    pattern = dataX[start]
    print("Seed:")
    print("\"", ''.join([int_to_char[value] for value in pattern]), "\"")
    sequence = []
    sys.stdout.write("Generated text - ")
    for i in range(1000):
        x = numpy.reshape(pattern, (1, len(pattern), 1))
```

```
        x = x / float(n_vocab)
        prediction = numpy.array(model_.predict(x, verbose=0))
        index = numpy.argmax(prediction)
        result = int_to_char[index]
        sequence.append(result)
        sys.stdout.write(result)
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
    f = open("generated_text.txt", "a")
    f.write('On epoch ' + str(epoch) + ':\n' + ''.join(sequence) + '\n')
    f.close()


class MonitorCallback(Callback):
    def __init__(self, epochs):
        super().__init__()
        self.epochs = epochs

    def on_epoch_end(self, epoch, logs=None):
        if epoch in self.epochs:
            generate_text(epoch, model)


filepath = "weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
save_best_only=True, mode='min')

model.fit(X, y, epochs=20, batch_size=128, callbacks=[checkpoint, MonitorCallback([1,
2, 3, 5, 7, 10, 14, 15, 16, 17, 18, 19, 20])])
```